Software Development at the German Aerospace Center: Role and Status in Practice

Lynn von Kurnatowski
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
lynn.kurnatowski@dlr.de

Tobias Schlauch
German Aerospace Center (DLR)
Braunschweig, Germany
tobias.schlauch@dlr.de

Carina Haupt
German Aerospace Center (DLR)
Berlin, Germany
carina.haupt@dlr.de

ABSTRACT

Software is an important innovation factor and an integral part of modern research. However, researchers are often faced with challenges in developing software because they do not have the necessary education and skills. The German Aerospace Center (DLR) established its *software engineering initiative* in 2005 to enable researchers to better meet these challenges. Continuous adaption and improvement of the supportive measures of the initiative require a good understanding of the current role and practice of software development at DLR. Therefore, we conducted a DLR-wide survey on research software development at DLR at the end of 2018.

In this paper, we present the results of this survey and identify possible improvements of the software engineering initiative activities. 773 DLR employees completed our survey and provided information about their academic background, programming experience, and software development practices. The results show that software development is a very relevant topic among the researchers at DLR but also a lack of applying software development best practices. Based on these results we conclude to further enhance the practical focus of our support activities as well as to raise the awareness for these practices to bring them into the daily work of DLR researchers.

KEYWORDS

data set, survey, research software, DLR

ACM Reference Format:
Lynn von Kurnatowski, Tobias Schlauch, and Carina Haupt. 2020. Software Development at the German Aerospace Center: Role and Status in Practice. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3387940.3392244

1 INTRODUCTION

DLR is a large research organization in Germany. It mainly conducts research in the domains space, aeronautics, energy, and transport. While software development plays a crucial role in the research activities of most scientific fields [1], it is regularly considered as simple tooling to automate tasks and is often based on an ad-hoc, code-and-fix approach without documentation, source code version control, or issue tracking.

ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7963-2/20/05.
https://doi.org/10.1145/3387940.3392244

In particular, software properties which contribute to its sustainability, such as re-usability, maintainability, and extensibility, are not prioritized or often not considered. All these properties are needed to re-use and maintain software artifacts over a longer period of time but are often not included in the curricula of (higher) education programs. In consequence, researchers of all fields are increasingly faced with software engineering challenges and have to maintain processes, for which they usually have received only little training.

For this reason, DLR started the *software engineering initiative* in 2005 to support their researchers in advancing their software engineering skills and in developing sustainable, high quality software.

As an initial step to evaluate the status quo of software development practice, we conducted a DLR-wide survey at the end of 2018 to get a detailed understanding of the heterogeneous landscape of software development at DLR and use this new knowledge to further improve the *software engineering initiative*.

The remaining paper presents the survey and its results as follows:

- We introduce the *software engineering initiative* and their support activities (Sect. 2).
- We describe the specific research questions and the method used (Sect. 3).
- We present the survey results (Sect. 4).
- We summarize the major findings and indicate future work directions (Sect. 5).

2 DLR SOFTWARE ENGINEERING INITIATIVE

In 2005, the *DLR software engineering initiative* was started to improve the quality and sustainability of research software at DLR [2]. The initiative is integrated into DLR's quality management policy via the *DLR software engineering directive*. The directive defines the overall software engineering policy and is mandatory for all DLR research institutes.

The *software engineering initiative* consists of different supplementary activities which focus on direct support of researchers developing software as part of their job at DLR:

1) The *DLR software engineering guidelines* [4] are part of DLR's software engineering policy. They define the desired software development and documentation practice at DLR. In addition, they include checklists helping researchers to evaluate the status of their software and to improve it iteratively.
2) Essential software development tools, such as version control systems and issue tracker, are provided centrally including a professional user support. The offering of such tools is intended to equip researchers properly to follow the software

engineering guidelines. It is important to point out that the survey has been conducted *before* the major switch from Subversion[1] to Git[2] and the collaboration platform GitLab[3].
3) Regular training courses are offered as part of DLR's education program. They focus on practical application of the software engineering guidelines and the centrally offered development tools.

In addition to the listed activities, we try to improve knowledge exchange and networking among DLR researchers by providing a moderated Wiki space as well as by performing annual knowledge exchange workshops.

3 METHOD

We analyzed the responses of the survey regarding the following questions:

1) How relevant is software development in the research activities?
2) What is the current state of practice with regard to software development?
3) What can we learn to improve the software engineering initiative activities?

Measures: The online survey consisted of 19 questions divided into six sections: demographics (4 questions), questions about coding as part of the job (4 questions), programming languages and tool usage (2 questions), documentation of software (2 questions), testing of software (3 questions), and citation of software (3 questions). The first question "*Do you write code as part of your job?*" acted as a filter. Participants who answered with "No" were directed to the last section of the survey. Responses to all questions were optional, except for the first one.

Participants: The target group included all DLR employees from all institutes. The survey was distributed to all DLR employees via email.

Data Analysis: We aggregated the data before publishing the data set to avoid disclosing the identity of the participants.
1) The participants were asked to assign themselves to one age group. To avoid identification of individual participants, we ensured that each age group consisted of at least 5 participants. Therefore, we had to combine groups.
2) Next, we assigned the 47 different institutes to their corresponding domains to avoid identification of individual participants using the institute affiliation.

No further anonymization steps were necessary, because we did not collect any further personal data.

The derived data set has been analyzed using Jupyter Notebook[4] and Pandas [3]. The Jupyter notebook containing this analysis code, the data set and the questionnaire has been published separately [5].

[1]https://subversion.apache.org/, accessed: 06.04.2020
[2]https://git-scm.com/, accessed: 06.04.2020
[3]https://about.gitlab.com/, accessed: 06.04.2020
[4]https://jupyter.org/, accessed: 06.04.2020

4 RESULTS

First, we show descriptive statistics describing the participant demographics. Then, we summarize the results concerning tool usage as well as documentation, testing, and software citation practices.

4.1 Sample Characteristics

4.1.1 General Participant Overview. The survey yielded 773 valid and complete responses from persons across the five domains. Most of the participants conduct research in space (44%) and aeronautics (33%), which are the two largest fields of research at DLR, followed by energy (10%), transport (6%) and other (8%).

Not only all fields of research, but also all age groups are represented in this survey. Persons with the age of 25 to 34 (41%) and 35 to 44 (35%) account for most of the survey participants (18-24 with 4%, 45-54 with 13% and 55 or older with 7%).

The majority of participants (79%) stated that they write code as part of their work, which shows that software development is an integral part of research at DLR. Only 21% reported that writing code is not part of their work.

4.1.2 Developer Overview. With the help of this survey, we wanted to get an understanding of how software is developed at DLR. Therefore, the following results took only into account those 612 participants who write code as part of their work.

The majority of the developers holds a master degree or doctorate (90%). Among the developers almost all disciplines were represented. However, the technical and natural science subjects, such as computer science (19%), aeronautical and manufacturing engineering (17%), mechanical engineering (16%), and physics (15%) constituted the largest groups.

To obtain a better overview about the general level of knowledge in software development, we also included a question concerning the years of software development experience the participants have. The range varied from absolute beginners with zero years of experience to professionals with 42 years of experience. The median of all developers was nine years of software development experience.

We also wanted to know by whom the software is used that they typically develop. This was rated by the participants on a scale from 0 (mostly me) to 5 (mostly other persons). The results show that most participants develop software for themselves (16%) or for a small group of other persons (1 with 22%, 2 with 26%). Only about 9% answered that they develop software mainly for other persons.

In addition to these general information, we asked how much time (in percent) each developer spends and would like to spend on (1) research, (2) software development, (3) management, and (4) other activities. Figure 1 shows the results. As shown in Figures 1(a),1(b), 1(c), and 1(d), developers want to spend more time on research and also more time on software development related activities. In contrast, developers want to spend less time on management and other activities (see Figures 1(e), 1(f) and 1(g), 1(h)).

4.2 Tool Usage

The diversity of research focuses is also reflected in the programming languages that are used. The most frequently used programming language is Python with about 23%, followed by C++ with about 14%, MATLAB with about 12% and C with about 11%.

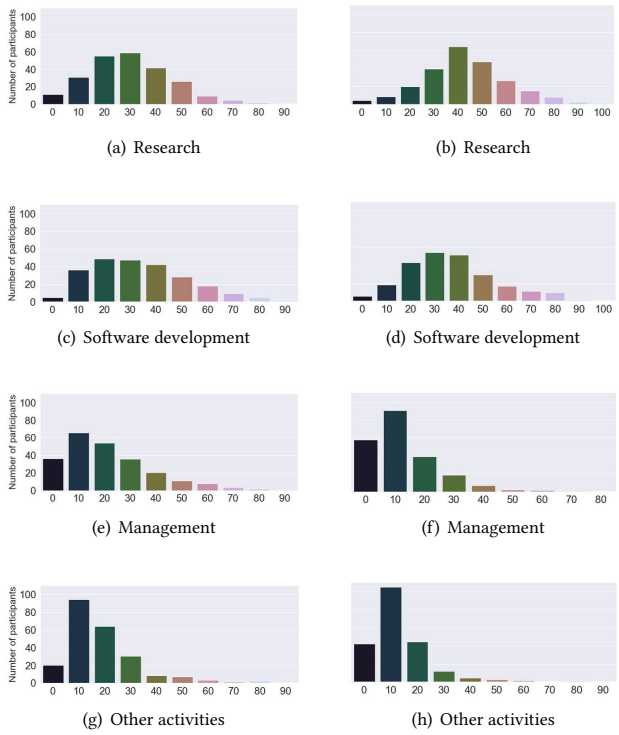

Figure 1: Actual time spent (left) (n=611) vs. preferred time spent (right) (n=606) on different activities

Among the version management tools, Git with 40% and Subversion with 37% are the most frequently used tools. 15% of the participants stated that they do not use any version control tool for software development.

4.3 Documentation

When developing software, not only writing source code is important but also the creation of documentation needs to be kept in mind. 88% of the respondents answered that they document their software. However, 12% of the participants also stated that they do not write any documentation at all.

In addition, we wanted to know what kind of documentation artifacts are usually created by the participants. As shown in Figure 2, the majority of participants equate documentation with writing code comments. `ReadMe` files or user manuals are only sometimes taken into consideration. The remaining data artifacts are rarely written.

4.4 Testing

In addition to documentation, software testing is another important part of software development. 97% of the participants stated that they test their software while 3% stated that they do not test their software at all. The majority of respondents tests their code regularly during development (83%) and before the software is given to someone else (65%) or the results based on it are published (72%).

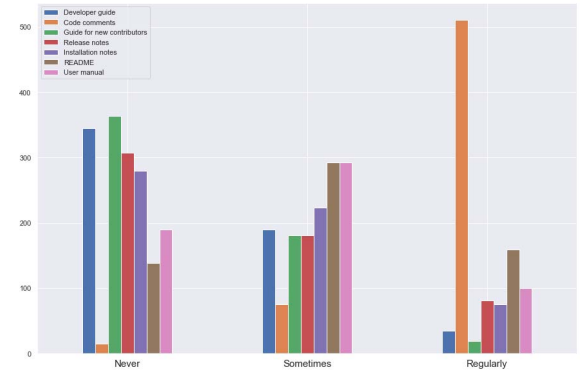

Figure 2: Question: "What kind of documentation do you write?" (n=606)

We further inspected the aspect of testing by examining how the participants test their software. The finding of this is that the majority of respondents tests their software manually (76%) rather than automatically (20%).

4.5 Software Citation

As the topic of research software and its role in a scientific context is becoming more and more important at DLR, we wanted to know how the participants deal with the topic of software citation. This covered questions whether they make their software citable, whether they reference used software in their own publications, and if so, how they do it.

The results show that about 68% of the 612 respondents do not care about making their developed software citable. But 56% of all 773 participants reference software in publications. Of those 56% who do, the main ways picked to reference software is using the software name or software in combination with a web link (23%). Only 9% of all participants reference software using a *Digital Object Identifier* (DOI).

5 SUMMARY AND FUTURE WORK

At the end of 2018, we conducted a DLR-wide online survey concerning research software development to find out about the relevance of software development in research and to assess the current state of practice. We analyzed the responses to identify possible improvements and optimizations of the supportive measures of the *DLR software engineering initiative*. Overall, the survey received 773 valid answers, among them 612 from DLR employees developing code in their current job position.

The survey showed a plausibly good coverage of the relevant target group. First of all, the response rate is only 9% when considering the total number of DLR employees (8444) at the end of 2018 [5]. However, the total number of DLR employees also includes potentially many persons that are clearly not concerned with software development. Typically, these persons are non-scientific staff working in context of infrastructure and support areas. When considering the descriptive statistics, we can see that all DLR research

[5]DLR Facts and Figures 2018, https://www.dlr.de/EN/organisation-dlr/media-and-documents/facts/facts-and-figures.html, accessed: 06.04.2020

fields are represented in the survey. In addition, the age distribution, the represented disciplines, and the typically achieved educational level indicate that the survey achieved a good coverage among DLR researchers.

5.1 Relevance of Software Development

Software development is a relevant topic among the researchers at DLR. The survey generally showed a good coverage of DLR researchers when taking into account parameters such as age distribution, level of education, discipline as well as associated DLR research field. In addition, it can be noted that the majority of participants develops software mainly for themselves or for a small number of other persons. This is distinctive for research as no scientist today can work without scripts or smaller software. Furthermore, results such as the diversity of used programming languages also indicate a broader relevance of software development at DLR. Finally, developers generally indicate their interest to invest more time in software development related activities. Overall, these aspects indicate that software development is an important tool for research work across all domains and disciplines at DLR.

5.2 Current State of Practice

The survey results show that a large variety of programming languages is used. However, we can identify a group of four programming languages which show a notable adoption. In contrast to the diversity of programming languages, the usage of version control systems mainly focuses on Git and Subversion. Interestingly, a slight majority of the survey respondents already used Git, although Subversion was DLR's official version control service at the time the survey has been conducted (Sect. 2).

Considering the current state of practice, we can identify a mismatch between recommended and the current practices among DLR researchers. On the one hand, the majority of software developing participants uses version control systems which can be considered fundamental tools when aiming for reproducible science. On the other hand, in the context of documentation, the results show a lack of writing commonly needed documentation artifacts such as user documentation except code comments. In addition, we can identify a clear lack of using test automation practices on a regular basis. Reasons for these findings might include that the survey respondents lack knowledge about these practices, are not aware of their relevance, or see no direct benefit of their usage. However, future studies are needed to identify the individual or structural reasons for these shortcomings.

Finally, software citation practices indicate plenty of room for improvements. Particularly, only a minority of survey respondents references software in research papers in a way that allows for reproducibility of scientific results. A major cause might be the general lack of recognizing software as an academic research result [1].

5.3 Identified Improvements

We need to further incorporate the recommendations of the DLR software engineering guidelines (Sect. 2) into the daily practice of DLR researchers.

On the one hand, we need to make researchers aware of the benefits of these practices and their relevance for reproducible science. In this context, it could be useful to cooperate with other activities at DLR, focusing on data management and open science, as we consider open science practices as a relevant driver for at least basic practices in research software development. In addition, they are more and more demanded by funding organizations and publishers.

On the other hand, we need to further enhance the practical focus of our activities, such as the offered training courses. For example, there is a need for more actionable materials which underpin the DLR software engineering guidelines (Sect. 2) and show their implementation using relevant practical examples, taking into account DLR's central collaboration platform GitLab and the widely used programming languages (Python, C, MATLAB, C++) at DLR. The recent introduction of GitLab and Git offers a good opportunity to improve the activities in this regard.

Finally, there is still a lack of recommendations concerning software citation, which needs to be addressed in future versions of the software engineering guidelines.

5.4 Future Work

As a first follow-up activity, we will conceptualize and prioritize the identified improvement ideas. Indeed, there are already ongoing activities in this context, for example, the redesigned training on version control, accompanying the GitLab introduction, as well as the recently introduced training with focus on publishing research software in accordance to open science principles. Secondly, we want to further investigate the impacts of our activities on a more fine-grained level. In particular, the consequences of the GitLab introduction are of special interest for us. Finally, we will also collaborate with the incubator platform HIFIS[6] in this regard which aims to establish similar support activities for research software development in the Helmholtz Association.

ACKNOWLEDGMENTS

The authors would like to thank all survey respondents for their participation and Katharina Dworatzyk for proofreading and commenting on the manuscript.

REFERENCES

[1] Alice Allen, Cecilia Aragon, Christoph Becker, Jeffrey Carver, Andrei Chis, Benoit Combemale, Mike Croucher, Kevin Crowston, Daniel Garijo, Ashish Gehani, Carole Goble, Robert Haines, Robert Hirschfeld, James Howison, Kathryn Huff, Caroline Jay, Daniel S. Katz, Claude Kirchner, Katie Kuksenok, Ralf Lämmel, Oscar Nierstrasz, Matt Turk, Rob van Nieuwpoort, Matthew Vaughn, and Jurgen J. Vinju. 2017. Engineering Academic Software (Dagstuhl Perspectives Workshop 16252). *Dagstuhl Manifestos* 6, 1 (2017), 1–20. https://doi.org/10.4230/DagMan.6.1.1

[2] Carina Haupt, Tobias Schlauch, and Michael Meinel. 2018. The Software Engineering Initiative of DLR – Overcome the obstacles and develop sustainable software. In *2018 ACM/IEEE International Workshop on Software Engineering for Science*. https://elib.dlr.de/120462/

[3] Wes McKinney. 2011. pandas: a Foundational Python Library for Data Analysis and Statistics. In *1st Workshop Python for High Performance and Scientific Computing (PyHPC 2011)*. Seattle.

[4] Tobias Schlauch, Michael Meinel, and Carina Haupt. 2018. DLR Software Engineering Guidelines. https://doi.org/10.5281/zenodo.1344612

[5] Lynn von Kurnatowski and Tobias Schlauch. 2020. *Role and practice of research software development at DLR.* https://doi.org/10.5281/zenodo.3611307

[6] https://software.hifis.net/, accessed:06.04.2020